Introduction :

To date, the majority of DEXs have been based on the Automated Market Maker (AMM) mechanism used by Uniswap, Curve, and others. On the other hand, it is an undeniable fact that as AMMs have become more popular, some serious problems have arisen. For example, Sandwich Attack is a pending issue for traders, and The concept of Loss versus Rebalancing (LVR) casts doubt on the sustainability. To begin with, Blockchain is a discrete system. Creating a system predicated on continuous operation poses significant challenges. It is also extremely difficult for a decentralized system to compete with a centralized system in terms of latency. To solve the above problems, Uniswap, for example, proposed UniswapX, which constructs a filler network using Dutch Auction. However, UniswapX also has three concerns: dynamic auction construction, centralization due to latency competition, and execution quality of the filler.

Current Problems :

1) Difficulty in building a dynamic auction on-chain

In RFQ systems such as UniswapX, 1inch, and CoWSwap, traders' orders take the form of bids by Fillers and Solvers via auctions. Auctions are considered a very important mechanism in Blockchain, where latency is a bottleneck, but it is extremely difficult to execute auctions on-chain. Suppose we consider a mechanism where the Bidder's Bid is included in the Block and it is executed for a certain period of time. In this case, the censorship resistance is the biggest concern. Even if an ordinary user makes a bid in the auction, it is the Builder or Proposer who includes the bid in the Block. However, if the auction is something from which some profit can be extracted, they will not actively include the user's transaction. It is also difficult to build an active auction model on-chain. For example, UniswapX defines an on-chain Decay Dutch Function, but since this function is based on timestamp, it ends up being a discrete auction. For example, consider the function pb = 100 - ts, where pb = 100 - 12s, since the Blocktime is 12 seconds in Ethereum. And if state = 1, the bidder would win at $88. However, it is not possible or meaningful for a bidder to win at $99 or $98. For bidders, latency can be a major issue for bidders because the amount available for bidding per Block is fixed in advance. These and other latency issues are discussed below.

2) Filler and Solver centralization due to latency competition

UniswapX and 1 inch Fusion use Dutch Auction, which tends to execute orders faster than English Auction. Dutch Auction is an auction system with a gradually decreasing price, which requires bidders to quickly place bids at their desired prices. This means that latency is a very important factor in bidding. For example, a typical Filler needs very high transmission speeds to send bid data from Tokyo to a central server in New York, and the physical distance to the server is another factor that cannot be ignored, as is the latency competition in the existing financial industry. Since Filler bases its bidding behavior on prices offered by Binance, Coinbase, and others, high latency is a major disadvantage. This high latency sensitivity can cause centralization problems for Filler Network. In other words, it could create a situation where only large players with fast communication networks and advanced communication technology can trade favorably. Of course, as long as Filler is always offering better prices, this cannot be a pressing issue, but it is an alarming one, such as a cartel in the event of an actual oligopoly. In fact, UniswapX transactions are executed by a very small number of fillers.

3) problems with execution quality of Filler and Solver

In UniswapX and 1inch Fusion, Filler and Solver have direct relative transactions such as Direct Fill, in addition to transactions through DEX Pools such as Uniswap and Curve. Here, we will take UniswapX as an example to delve deeper into Filler's execution quality. Currently, the UniswapX filler is extremely centralized, with two to three addresses handling UniswapX transaction. UniswapX has a diverse set of parameters that users must evaluate when trading, and it is very difficult to scrutinize them. Certainly, parameters are specified as given from the UniswapX UI, so there is not much effort required from the user. On the other hand, there is no way to determine whether the set Parameter is appropriate or not. Although it is expected that users can enjoy better execution quality through DEX's Pool, Filler and Solver execute uninformed flows that are not appropriate by direct fill. We have also compared DEX prices with respect to Filler and Solver's execution quality. However, if more uninformed flow is executed directly by Filler and Solver, liquidity providers will be exposed to more toxic flow. In this case, liquidity providers would suffer further losses and they would exit. Therefore, it can be pointed out that, especially in DEX pools where the CEX takes the lead in determining token prices, it may become difficult to obtain prices on-chain based on AMM as it has been in the past in the long run.

Solution : Frequent Batch Auction on SUAVE

What is a Frequent Batch Auction?

Implementation :

[

Untitled-2024-01-07-1151

1920×1277 81.1 KB

](https://collective.flashbots.net/uploads/default/original/2X/4/47ddf0a1547f47025c20f30fe9491e6bad8de96b.jpeg)

Here is a brief description of the implementation of Frequent Batch Auction in SUAVE.

(1) Traders and Market Makers send bid and cancel preferences to SUAVE's Auction Contract.

(2) SUAVE's Auction Contract stores the received bid and cancel information in a confidential data store.

(3) Operator executes the Auction Contract based on a fixed interval of τ time.

(4) Auction Contract acquires the information of each bid and cancellation stored in the confidential data store.

(5) In the auction contract, the bid and ask demand and supply curves are determined by functions defined on the MEVM, and the orders are sorted accordingly.

(6) Bundle is generated from the sorted orders and sent to Builder and Relayer.

Benefits of using Frequent batch Auction

1) No need for dynamic auctions

Frequent Batch Auction uses uniform-price sealed-bid double auctions, eliminating the need for dynamic auctions such as English and Dutch Auctions. Iin the Frequent Batch Auction, bidders' bid information must be kept confidential until the end of the τ time. In this case, SUAVE's confidential data store is used to prevent traders' bid information from being seen by other traders. In addition, there are various problems in active auction, such as strategic resistance in Dutch Auction, and agility in bidding time in English Auction. Other important issues are organized below.

2) Prevention of centralization of Filler and Solver through latency competition

[

スクリーンショット 2024-01-20 2.49.05

1890×670 19.4 KB

](https://collective.flashbots.net/uploads/default/original/2X/5/5688c8b11ebaa668fdcc413c6df82a892ce0e1ce.png)

In dynamic auctions, latency is a very important factor in bid competition. In the Frequent Batch Auction, on the other hand, bids by traders are processed after a certain time, τ, so latency is not an issue and advanced communication technology and equipment are not required. This reduces the risk of oligopoly by low latency builders such as fillers and solvers in the block builder market. (In Frequent Batch Auctions, it may be more appropriate to refer to them as Market Makers, who are relative to users' trades, rather than Solvers or Fillers, who execute users' intents.)

Also, by following the Auction Contract defined on SUAVE, arbitrary Bundle and Block construction by Searchers and Builders, as has been observed in MEV-Boost, can be prevented. In terms of latency, Frequent Batch Auction allows any actor to bid or cancel with respect to (τ0 , τs) regardless of latency. For (τs , τh), only low latency actors can respond, but since these orders are processed by auction, it is highly unlikely that they will be processed by the old quote of a latency actor. In (τh , τ), not all actors can react, so they are not taken into account. Therefore, the probability that an actor with latency will be sniped is (τs-τh)/τ, which is much smaller than the nearly 100% probability of being sniped in the past.

3) Improvement of Execution Quality

In the upper part, we confirmed that the probability that an actor with latency is sniped by a low latency Informed Trader is (τs-τh)/τ. Then, during the period (τs , τh), a latency actor is not necessarily sniped by an old quote. Consider a simple order book situation. Suppose an actor with latency executes a limit order to sell at $1001 when the current price of a token is $1000. Next, if some signal indicates that the token price is $1005, Informed trader A with low latency will try to snipe the order. However, in the Frequent Batch Auction, sending a buy order at $1001 at this time does not necessarily mean that it will match the earlier order. However, if another informed trader B sends a buy order at $1002, the earlier order will be matched with this B order. Since there is a bidding war between informed traders, it is desirable to improve the execution quality of buy and sell orders by latency actors.

4) Reduction of Loss versus Rebalancing

Frequent Batch Auction is expected to reduce Loss versus Rebalancing, which is currently the biggest problem in AMM. Assume that the ETH price is $2000 and Market Maker provides liquidity to the 2001 USDC tick. Let us assume that the ETH price on Binance now rises to 2005 USDC. In this case, an informed trader such as Searcher or Builder comes to DEX and tries to "Snipe" the liquidity provided. If the liquidity provider is not an advantageous actor, 100% of the provided liquidity will be sniped. This is because transactions called CEX-DEX arbitrage are very likely to be included at the top of the Block because of the high extraction value of the transaction itself. Therefore, even if a typical liquidity provider were to withdraw its own liquidity, the cancel order would be included after the snip transaction, and would end up being exposed to toxic flow. Even if block builder itself provides liquidity, it does not necessarily avoid sniping. Suppose that the block building ability of the top N block builders is balanced. In other words, the sorting algorithm, private order flow, latency, and other factors that differentiate block builder A from block builder B are assumed to be comparable. In this case, the probability that block builder A can build a Block is 1 / N. Conversely, the probability that another builder will build a Block, i.e., that the liquidity offered by builder A will be sniped, is N - 1 / N.

On the other hand, in the Frequent Batch Auction,the informed trader is unable to fully capitalize on the aforementioned advantageous points, and the profit of the arbitrager in the case of sniping is very small due to the improvement of the auction quality as confirmed in 3. These points are expected to play an important contribution to the reduction of LVR.

5) Reduction of Front running and Sandwich Attack

In the Frequent Batch Auction, which utilizes SUAVE's confidential data store, accepted orders are not disclosed until the end of the bid period. The accepted orders are used to form a demand/supply curve, specifically to recreate the same situation as in the orderbook, only with the use of confidential order information. Therefore, users are not able to know the orders of other users, and since the processing of orders follows the pro-rata method, the order in which the orders arrive within the same bid period is completely unimportant. This removes problems caused by order sequence, such as front running, back running, and sandwich attack.

Future work :

1) Sustainability of Liquidity Provider

Excessive latency competition and the resulting exit of liquidity providers are serious problems in the existing Central Limit Order Book and AMM. However, further research is needed to determine whether the Frequent Batch Auction can truly provide Liquidity Providers with an environment in which they can build a sustainable Market Making Strategy. For example, since the Frequent Batch Auction was proposed, there has been no full-scale introduction except for the Taiwan stock exchange. The Taiwan stock exchange has also shifted to a continuous limit order book after 2021. In addition, it has been pointed out that the Frequent Batch Auction method may make it difficult for the market maker strategy, which worked at least in the Zaraba method, to generate profits commensurate with the risk involved. While many apply the best market making strategy to CLOB in the first place, it is also important to consider whether there is a market maker strategy suitable for the batch auction method itself.

2) Handling multiple rollups

A possible design and implementation of a multi-rollup frequent batch auction using SUAVE and a shared sequencer is to aggregate the Intents of multiple rollups by using SUAVE as a mempool for multiple rollups. SUAVE can aggregate the Intents of multiple rollups by taking on the role of a mempool of multiple rollups and canceling out the transactions contained in the Intents. For example, consider the case where User A sends to SUAVE the Intents to swap 1 ETH from 10000 USDC in Arbitrum and User B sends to SUAVE the Intents to swap 1 ETH to 10000 USDC in Optimism. In this case, the two Intents are offset within SUAVE, and a transaction in which User B bridges 1 ETH from Optimism to Arbitrum and sends 1 ETH to User A and a transaction in which User A bridges 10000 USDC from Arbiturm to Optimism User A bridges 10000 USDC from Arbiturm to Optimism and sends 1 ETH to User B. Then, SUAVE generates a mega block that allows Atomic execution of the two transactions. The megablock generated by SUAVE is then sent to the shared sequencer to be sent to each rollup for atomic execution of the two transactions. The problem with this mechanism is the transactional issue of how to handle transactions that cannot be offset, and the technical issue of how to guarantee that transactions reflecting offsetting will be executed atomically.

3) Dealing with Multi tokens

So far, we have proposed a Batch Auction for a single token pair. On the other hand, there are many reasons why it should be possible to trade many tokens all in one system: Frequent Batch Auctions are traded when traders' supply and demand match. However, even taking Ethereum's WETH-USDC 0.05% pool as an example, there are not many orders being executed per block, especially in normal times, and it is quite possible that a batch period per block time would not create an overlap in the order book, let alone a supply/demand curve, if the batch period is set for each block time. On the other hand, if we consider the case where a trader wants to execute trades A->B, B->C, C->A, etc., there is a possibility that a supply-demand match will occur. This is what is called a ring transaction.

Certainly, it is impossible to find a price that would optimize the prices of all token pairs. However, there are projects, such as CoW Swap, that match multiple token pairs for Batch processing. We will continue to research optimal matching strategies.

Reference

Bid, ask and transaction prices in a specialist market with heterogeneously informed traders

The High-Frequency Trading Arms Race: Frequent Batch Auctions as a Market Design Response

Improved Price Oracles: Constant Function Market Makers

Automated Market Making and Loss-Versus-Rebalancing